PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ^{6} : **G06F 12/08**	**A1**	**(11) International Publication Number:** **WO 99/50752** **(43) International Publication Date:** 7 October 1999 (07.10.99)

(21) International Application Number: PCT/US99/06501

(22) International Filing Date: 24 March 1999 (24.03.99)

(30) Priority Data:
09/053,386 31 March 1998 (31.03.98) US

(71) Applicant *(for all designated States except US)*: INTEL CORPORATION [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

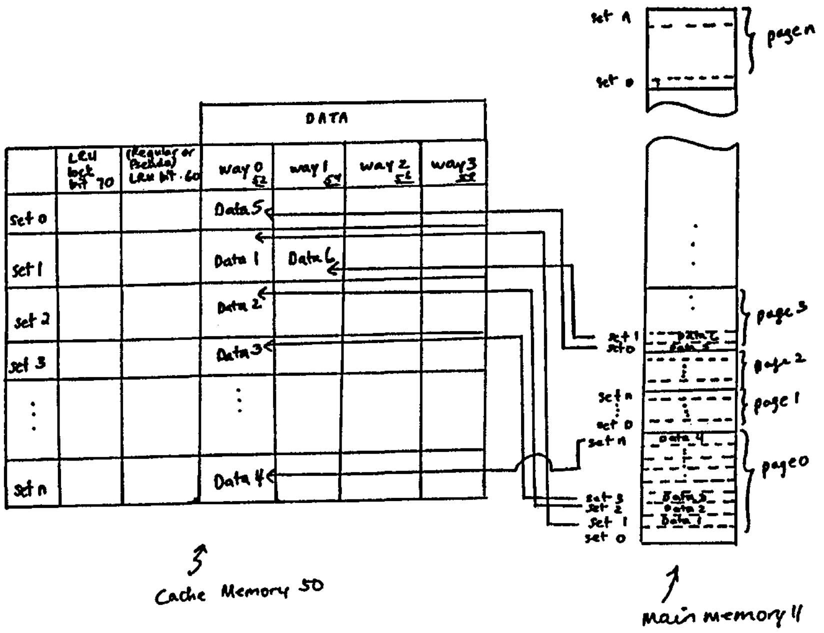
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: PALANCA, Salvador [ES/US]; 2400 Natoma Station Drive #128, Folsom, CA 95630 (US). COORAY, Niranjan, L. [LK/US]; 373 Hansen Circle, Folsom, CA 95630 (US). NARANG, Angad [IN/US]; 2330 Vehicle Drive #3, Rancho Cordova, CA 95670 (US). PENTKOVSKI, Vladimir [RU/US]; 221 Luna Circle, Folsom, CA 95630 (US). TSAI, Steve [US/US]; 3466 Data Drive #936, Rancho Cordova, CA 95670 (US).

(74) Agents: CORTES, Roland, B. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: SHARED CACHE STRUCTURE FOR TEMPORAL AND NON–TEMPORAL INSTRUCTIONS



(57) Abstract

A method and system for providing cache memory management. The system comprises a main memory (11), a processor coupled to the main memory, and at least one cache memory (50) coupled to the processor for caching of data. The at least one cache memory has at least two cache ways (50), each comprising a plurality of sets (50). Each of the plurality of sets has a bit (50) which indicates whether one of the at least two cache ways contains non–temporal data. The processor accesses data from one of the main memory or the at least one cache memory.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

SHARED CACHE STRUCTURE FOR

TEMPORAL AND NON-TEMPORAL INSTRUCTIONS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates in general to the field of processors, and in particular, to a technique of providing a shared cache structure for temporal and non-temporal instructions.

2. Description of the Related Art

The use of a cache memory with a processor facilitates the reduction of memory access time. The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in the fast cache memory, the average memory access time will approach the access time of the cache. To achieve the maximum possible speed of operation, typical processors implement a cache hierarchy, that is, different levels of cache memory. The different levels of cache correspond to different distances from the processor core. The closer the cache is to the processor, the faster the data access. However, the faster the data access, the more costly it is to store data. As a result, the closer the cache level, the faster and smaller the cache.

The performance of cache memory is frequently measured in terms of its hit ratio. When the processor refers to memory and finds the word in cache, it is said to produce a hit. If the word is not found in cache, then it is in main memory and it counts as a miss. If a miss occurs, then an allocation is made at the entry indexed by the access. The access can be for loading data to the processor or storing data from the processor to memory. The cached information is retained by the cache memory until it is no longer needed, made invalid or replaced by other data, in which instances the cache entry is de-allocated.

In processors implementing a cache hierarchy, such as the Pentium Pro™ processors which have an L1 and an L2 cache, the faster and smaller L1 cache is located closer to the processor than the L2 cache. When the processor requests cacheable data, for example, a load instruction, the request is first sent to the L1 cache. If the requested data is in the L1 cache, it is provided to the processor. Otherwise, there is an L1 miss and the request is transferred to the L2 cache. Likewise, if there is an L2 cache hit, the data is passed to the L1 cache and the processor core. If there is an L2 cache miss, the request is transferred to main memory. The main memory responds to the L2 cache miss by providing the requested data to the L2 cache, the L1 cache, and to the processor core.

The type of data that is typically stored in cache includes active portions of programs and data. When the cache is full, it is necessary to replace existing lines of stored data in the cache memory to make room for newly requested lines of data. One such replacement technique involves the use of the least recently used (LRU) algorithm, which replaces the least recently used line of data with the newly requested line. In the Pentium Pro™ processors, since the L2 cache is larger than the L1 cache, the L2 cache typically stores everything in the L1 cache and some additional lines that have been replaced in the L1 cache by the LRU algorithm.

U.S. Patent Application serial number 08/767,950 filed December 17, 1996, entitled "Cache Hierarchy Management" discloses a technique for allocating cache memory through the use of a locality hint associated with an instruction. When a processor accesses memory for transfer of data between the processor and the memory, that access can be allocated to the various levels of cache, or not allocated to cache memory at all, according to the locality hint associated with the instruction. Certain instructions are used infrequently. For example, non-temporal prefetch instructions preload data which the processor does not require immediately, but which are anticipated to be required in the near future. Such data is typically used only once or will not be reused in the immediate future, and is termed "non-temporal data". Instructions that are frequently used are termed "temporal data". For non-temporal data, since the data is used infrequently, optimal performance dictates that the cached application code and data not be overwritten by this infrequently used data. U.S. Application serial number 08/767,950 solves this problem by providing a buffer, separate from the cache memory, for storing the infrequently used data, such as non-temporal prefetched data. However, the use of an extra, separate buffer is expensive both in terms of cost and space.

Accordingly, there is a need in the technology for providing a shared cache structure for temporal and non-temporal instructions, which eliminates the use of a separate buffer.

BRIEF SUMMARY OF THE INVENTION

A method and system for providing cache memory management. The system comprises a main memory, a processor coupled to the main memory, and at least one cache memory coupled to the processor for caching of data. The at least one cache memory has at least two cache ways, each comprising a plurality of sets. Each of the plurality of sets has a bit which indicates whether one of the at least two cache ways contains non-temporal data. The processor accesses data from one of the main memory or the at least one cache memory.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is illustrated by way of example, and not limitation, in the figures. Like reference indicate similar elements.

Figure 1 illustrates a circuit block diagram of one embodiment of a computer system which implements the present invention, in which a cache memory is used for data accesses between a main memory and a processor of the computer system.

Figure 2 is a circuit block diagram of a second embodiment of a computer system which implements the present invention, in which two cache memories are arranged into cache memory levels for accessing data between a main memory and a processor(s) of the computer system.

Figure 3 is a block diagram illustrating one embodiment of the organizational structure of the cache memory in which the technique of the present invention is implemented.

Figure 4 is a table illustrating the cache management technique, according to one embodiment of the present invention.

Figures 5A and 5B illustrate one example of the organization of a cache memory prior to and after temporal instruction hits way 2 of cache set 0, according to one embodiment of the present invention.

Figures 6A and 6B illustrate another example of the organization of a cache memory prior to and after temporal instruction hits way 2 of cache set 0, according to one embodiment of the present invention.

Figures 7A - 7D illustrate a example of the organization of a cache memory prior to and after a non-temporal instruction hits way 2 of cache set 0, according to one embodiment of the present invention.

Figures 8A - 8D illustrate another example of the organization of a cache memory prior to and after a non-temporal instruction hits way 2 of cache set 0, according to one embodiment of the present invention.

Figures 9A and 9B illustrate one example of the organization of a cache memory prior to and after a temporal instruction miss to cache set 0, according to one embodiment of the present invention.

Figures 10A - 10D illustrate an example of the organization of a cache memory prior to and after a non-temporal instruction miss to cache set 0, according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

A technique is described for providing management of cache memories, in which cache allocation is determined by data utilization. In the following description, numerous specific details are set forth, such as specific memory devices, circuit diagrams, processor instructions, etc., in order to provide a thorough understanding of the present invention. However, it will be appreciated by one skilled in the art that the present invention may be practiced without these specific details. In other instances, well known techniques and structures have not been described in detail in order not to obscure the present invention. It is to be noted that a particular implementation is described as a preferred embodiment of the present invention, however, it is readily understood that other embodiments can be designed and implemented without departing from the spirit and scope of the present invention. Furthermore, it is appreciated that the present invention is described in reference to a serially arranged cache hierarchy system, but it need not be limited strictly to such a hierarchy.

Referring to Figure 1, a typical computer system is shown, wherein a processor 10, which forms the central processing unit (CPU) of the computer system is coupled to a main memory 11 by a bus 14. The main memory 11 is typically comprised of a random-access-memory and is usually referred to as RAM. Subsequently, the main memory 11 is generally coupled to a mass storage device 12, such as a magnetic or optical memory device, for mass storage (or saving) of information. A cache memory 13 (hereinafter also referred simply as cache) is coupled to the bus 14 as well. The cache 13 is shown located between the CPU 11 and the main memory 11, in order to exemplify the functional utilization and transfer of data associated with the cache 13. It is appreciated that the actual physical placement of the cache 13 can vary depending on the system and the processor architecture. Furthermore, a cache controller 15 is shown coupled to the cache 13 and the bus 14 for controlling the operation of the cache 13. The operation of a cache controller, such as the controller 15, is known in the art and, accordingly, in the subsequent Figures, cache controllers are not illustrated. It is presumed that some controller(s) is/are present under control of the CPU 10 to control the operation of cache(s) shown.

In operation, information transfer between the memory 11 and the CPU 10 is achieved by memory accesses from the CPU 10. When cacheable data is currently or shortly to be accessed by the CPU 10, that data is first allocated in the cache 13. That is, when the CPU 10 accesses a given information from the memory 11, it seeks the information from the cache 13. If the accessed data is in the cache 13, a "hit" occurs. Otherwise, a "miss" results and cache allocation for the data is sought. As currently practiced, most accesses (whether load or store) require the allocation of the cache 13. Only uncacheable accesses are not allocated in the cache.

Referring to Figure 2, a computer system implementing a multiple cache arrangement is shown. The CPU 10 is still coupled to the main memory 11 by the bus 14 and the memory 11 is then coupled to the mass storage device 12. However, in the example of Figure 2, two separate cache memories 21 and 22 are shown. The caches 21-22 are shown arranged serially and each is representative of a cache level, referred to as Level 1 (L1) cache and Level 2 (L2) cache, respectively. Furthermore, the L1 cache 21 is shown as part of the CPU 10, while the L2 cache 22 is shown external to the CPU 10. This structure exemplifies the current practice of placing the L1 cache on the processor chip while lower level caches are placed external to it, where the lower level caches are further from the processor core. The actual placement of the various cache memories is a design choice or dictated by the processor architecture. Thus, it is appreciated that the L1 cache could be placed external to the CPU 10.

Generally, CPU 10 includes an execution unit 23, register file 24 and fetch/decoder unit 25. The execution unit 23 is the processing core of the CPU 10 for executing the various arithmetic (or non-memory) processor instructions. The register file 24 is a set of general purpose registers for storing (or saving) various information required by the execution unit 23. There may be more than one register file in more advanced systems. The fetch/decoder unit 25 fetches instructions from a storage location (such as the main memory 11) holding the instructions of a program that will be executed and decodes these instructions for execution by the execution unit 23. In more advanced processors utilizing pipelined architecture, future instructions are prefetched and decoded before the instructions are actually needed so that the processor is not idle waiting for the instructions to be fetched when needed.

The various units 23-25 of the CPU 10 are coupled to an internal bus structure 27. A bus interface unit (BIU) 26 provides an interface for coupling the various units of CPU 10 to the bus 14. As shown in Figure 2, the L1 cache is coupled to the internal bus 27 and functions as an internal cache for the CPU 10. However, again it is to be emphasized that the L1 cache could reside outside of the CPU 10 and coupled to the bus 14. The caches can be used to cache data, instructions or both. In some systems, the L1 cache is actually split into two sections, one section for caching data and one section for caching instructions. However, for simplicity of explanation, the various caches described in the Figures are shown as single caches with data, instructions and other information all referenced herein as data. It is appreciated that the operations of the units shown in Figure 2 are known. Furthermore it is appreciated that the CPU 10 actually includes many more components than just the components shown. Thus, only those structures pertinent to the understanding of the present invention are shown in Figure 2. In one embodiment, the invention is utilized in systems having data caches. However, the invention is applicable to any type of cache.

It is also to be noted that the computer system may be comprised of more than one CPU (as shown by the dotted line in Figure 2). In such a system, it is typical for multiple CPUs to share the main memory 11 and/or mass storage unit 12. Accordingly, some or all of the caches associated with the computer system may be shared by the various processors of the computer system. For example, with the system of Figure 2, L1 cache 21 of each processor would be utilized by its processor only, but the main memory 11 would be shared by all of the CPUs of the system. In addition, each CPU has an associated external L2 cache 22.

The invention can be practiced in a single CPU computer system or in a multiple CPU computer system. It is further noted that other types of units (other than processors) which access memory can function equivalently to the CPUs described herein and, therefore, are capable of performing the memory accessing functions similar to the described CPUs. For example, direct memory accessing (DMA) devices can readily access memory similar to the processors described herein. Thus, a computer system having one processor (CPU), but one or more of the memory accessing units would function equivalent to the multiple processor system shown described herein.

As noted, only two caches 21-22 are shown. However, the compute system need not be limited to only two levels of cache. It is now a practice to utilize a third level (L3) cache in more advanced systems. It is also the practice to have a serial arrangement of cache memories so that data cached in the L1 cache is also cached in the L2 cache. If there happens to be an L3 cache, then data cached in the L2 cache is typically cached in the L3 cache as well. Thus, data cached at a particular cache level is also cached at all higher levels of the cache hierarchy.

Figure 3 is a block diagram illustrating one embodiment of the organizational structure of the cache memory in which the technique of the present invention is implemented. In general, there are "x" sets in a cache structure, "y" ways per set (where $y \geq 2$), and where each way contains one data entry or one cache line. The invention provides an LRU lock bit which indicates whether any one of the ways within that set contains non-temporal (NT) data. If so, the regular or pseudo LRU bits will be updated to point to the NT data. There are also "z" regular or pseudo LRU bits per set. Unless the LRU lock bit is set, the regular or pseudo LRU bits point to the way within the set in accordance with the least recently used technique implemented. The number of regular or pseudo-LRU bits per set varies depending on the number of ways per set and the LRU (regular or pseudo) technique implemented.

In the embodiment as shown, the cache 50 is organized as a four-way set associative cache. In the example of Figure 3, each page is shown as being equal to one-fourth the cache size. In particular, the cache 50 is divided into four ways (for example, way 0 (52), way 1 (54), way 2 (56) and way 3 (58)) of equal size and main memory 11 (see also Figures 1 and 2) is viewed as divided into pages (e.g., page 0 - page n). In another embodiment, each page may be larger or smaller than the cache size. The organizational structure of cache 50 (as shown in Figure 3) may be implemented within the cache 13 of Figure 1, the L1 cache and/or L2 cache 22 of Figure 2.

The cache 50 also includes an array of least recently used (LRU) bits 60_0 - 60_n each of which points to the way within a set with the least recently used data (or NT data, if a biased LRU technique is implemented). Such listing is performed in accordance with an LRU technique under the control of the cache controller 15, to determine which cache entry to overwrite in the event that a cache set is full. The LRU logic (not shown) keeps track of the cache locations within a set that have been least recently used. In one embodiment, an LRU technique that strictly keeps track of the least-recently used directory algorithm may be implemented. In one alternate embodiment, a pseudo-LRU algorithm which makes a best attempt at keeping track of the least recently used directory element, is implemented. For discussion purposes, the bits 60_0 - 60_n will be referred to as LRU bits 60_0 - 60_n, while the array of LRU bits 60_0 - 60_n will be referred to as LRU bits 60.

The cache 50 further includes an array of LRU lock bits 70_0 - 70_n which indicates whether any of the ways 52, 54, 56, 58 within a given set contains data that should not pollute the cache 50 (i.e., data with infrequent usage), as described in detail in the following sections.

Figure 4 is a table illustrating the cache management technique in accordance with the principles of the present invention. The invention utilizes the array of LRU lock bits 70_0 - 70_n to indicate whether any of the corresponding cached data is streaming or non-temporal, and as such, would be the first entry to be replaced upon a cache miss to the corresponding set. In one embodiment, the LRU lock bit 70, when set to 1, indicates that the corresponding set has an entry that is non-temporal. If the LRU lock bit 70 is cleared, upon a cache hit by a temporal instruction, the corresponding LRU bit(s) 60 is(are) updated in accordance with the LRU technique implemented (see item 1 of Figure 4) and the associated LRU lock bit is not updated. However, if the LRU lock bit 70 is already set to 1 (indicating that the corresponding set has a non-temporal instruction), the LRU lock bit 70 is not updated, and the LRU bit 60 is not updated (see item 2).

In the case of a cache hit by a non-temporal instruction, the LRU bit 60 and the LRU lock bit 70 are not updated, regardless of the status of the LRU lock bit 70 (see item 3). In an alternate embodiment, as controlled through a mode bit in a control register in the L1 cache controller, cache hits by a streaming or non-temporal instructions force the LRU bits to the way that was hit (see item 4). In addition, the LRU lock bit 70 is set to 1. In this embodiment, the data hit by the streaming or non-temporal instruction will be the first to be replaced upon a cache miss to the corresponding set.

Upon a cache miss by a temporal instruction, the LRU lock bit is cleared and the LRU bit 60 is updated (item 5) based on a pseudo LRU technique. However, upon a cache miss by a streaming or non-temporal instruction, the LRU lock bit 70 is set to 1 and the corresponding LRU bit 60 is not updated (item 6).

Examples of each of the items provided in the table of Figure 4 will now be discussed. Figures 5A and 5B illustrate one example of the organization of a cache memory prior to and after temporal instruction hits way 2 of cache set 0. This example corresponds to item 1 of Figure 4. Here, LRU lock bit 70_0 had been previously cleared for cache set 0, and since the cache set 0 was hit by a temporal instruction, the LRU lock bit 70_0 is not updated. However, the LRU bit 60_0 is updated in accordance with the LRU technique implemented. In the example, it is assumed that the pseudo LRU technique indicates that way 3 is the least recently used entry.

Figures 6A and 6B illustrate another example of the organization of a cache memory prior to and after temporal instruction hits way 2 of cache set 0. This example corresponds to item 2 of Figure 4. Here, LRU lock bit 70_0 had been previously set for cache set 0, indicating that the corresponding set contains non-temporal data. Accordingly, neither the LRU lock bit 70_0 nor the LRU bit 60_0 is updated.

Figures 7A - 7D illustrate an example of the organization of a cache memory prior to and after a non-temporal instruction hits way 2 of cache set 0. This example corresponds to item 3 of Figure 4 and may be implemented by setting a mode bit located in the L1 cache controller to zero (see Figure 4). In the first case (Figures 7A and 7B), LRU lock bit 70_0 had been previously cleared for cache set 0. In this embodiment, a non-temporal cache hit does not update the LRU lock bit 70. Accordingly, since the cache set 0 was hit by a non-temporal instruction, neither the LRU lock bit 70_0 nor the LRU bit 60_0 is updated. In the second case (Figures 7C and 7D), LRU lock bit 70_0 had been previously set for cache set 0, indicating that the corresponding set contains non-temporal data. Accordingly, neither the LRU lock bit 70_0 nor the LRU bit 60_0 is updated.

Figures 8A - 8D illustrate another example of the organization of a cache memory prior to and after a non-temporal instruction hits way 2 of cache set 0. This example corresponds to item 4 of Figure 4 and may be implemented by setting the mode bit located in the L1 cache controller to one (see Figure 4). In the first case (Figures 8A and 8B), LRU lock bit 70_0 had been previously cleared for cache set 0. In this example of an alternate embodiment to that example shown in Figures 7A-7D, a non-temporal cache hit updates the LRU lock bit 70. Accordingly, as shown in Figure 8A, since the cache set 0 was hit by a non-temporal instruction, the LRU lock bit 70_0 is updated (set to 1), as shown in Figure 8B. In addition, the LRU bits 60_0 are updated to indicate the way that was hit. In the case where LRU lock bit 70_0 had been previously set for cache set 0 (Figures 8C and 8D), the LRU lock bit 70_0 remains set to 1. In addition, the LRU bits 60_0 are forced to point to the way within the set that was hit.

Figures 9A and 9B illustrate one example of the organization of a cache memory prior to and after a temporal instruction miss to cache set 0. This example corresponds to item 5 of Figure 4. Here, LRU lock bit 70_0 had been previously set for cache set 0, and since there is a miss by a temporal instruction targeting set 0, the LRU lock bit 70_0 is cleared for that set, upon replacing the temporal miss in the cache. However, the LRU bit 60_0 is updated in accordance with the LRU technique implemented. In the example, the pseudo LRU technique indicates that way 3 is the least recently used entry.

Figures 10A - 10B illustrate an example of the organization of a cache memory prior to and after a non-temporal instruction miss to cache set 0. This example corresponds to item 6 of Figure 4. In this case, LRU lock bit 70_0 had been previously cleared for cache set 0. Since there is a non-temporal miss to cache set 0, the LRU lock bit 70_0 is set and the LRU bits 60_0 remain the same, in order to point to the non-temporal data in the corresponding set 0.

By implementing the apparatus and method of the present invention, a shared cache structure for managing temporal and non-temporal instructions, which minimizes data pollution in cache or cache hierarchy is provided. Implementation of the present invention also eliminates the use of a separate buffer, making its implementation both cost effective and efficient.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS

What is claimed is:

1. A computer system for providing cache memory management, comprising:

a main memory;

a processor coupled to said main memory;

at least one cache memory coupled to said processor, said at least one cache memory having at least two cache ways each comprising a plurality of sets, each of said plurality of sets having a first bit indicative of whether one of said at least two cache ways contains non-temporal data;

wherein said processor accesses data from one of said main memory or said at least one cache memory.

2. The computer system of Claim 1, wherein said at least one cache memory further comprises a second bit indicative of an order of a data entry in a corresponding way.

3. The computer system of Claim 2, wherein said order is indicative of whether said data entry is a least recently used entry with respect to other entries.

4. The computer system of Claim 1, wherein said first bit is set to indicate that an associated way contains non-temporal data.

5. The computer system of Claim 1, wherein said first bit is cleared to indicate that an associated way contains temporal data.

6. The computer system of Claim 2, further comprising cache control logic coupled to said at least one cache memory and said processor, for controlling said at least one cache memory.

7. The computer system of Claim 6, wherein said processor receives an instruction for accessing data, said processor determining if said data is located in said at least one cache memory, if so, accessing said data from said at least one cache memory, otherwise, accessing said data from said main memory.

8. The computer system of Claim 7, wherein if said data is accessed from said at least one cache memory, said cache control logic determines if said data is temporal, if so, updating an order of said second bit corresponding to said way that is being accessed, otherwise leaving said order unchanged.

9. The computer system of Claim 8, wherein said first bit corresponding to said way is unchanged.

10. The computer system of Claim 7, wherein if said data is accessed from said at least one cache memory, said cache control logic configures said first bit to indicate that said accessed data is non-temporal, said cache control logic further updating said order of said second bit.

11. The computer system of Claim 7, wherein if said data is accessed from said main memory, said cache control logic determines if said data is non-temporal, if so, configuring said first bit to indicate that said accessed data is non-temporal, said cache control logic leaving unchanged said order of said second bit.

12. The computer system of Claim 11, wherein if said cache control logic determines that said data is temporal, said cache control logic configures said first bit to indicate that said accessed data is temporal, said cache control logic updating said order of said second bit.

13. In a computer system, a method of allocating cache memories based on a pattern of accesses for data utilized by a processor, comprising:

providing a main memory;

providing a processor coupled to said main memory;

providing at least one cache memory coupled to said processor, said at least one cache memory having at least two cache ways each comprising a plurality of sets, each of said plurality of sets having a first bit indicative of whether one of said at least two cache ways contains non-temporal data,

accessing, by said processor, data from one of said main memory or said at least one cache memory.

14. The method of Claim 13, wherein said at least one cache memory further comprises a second bit indicative of an order of a data entry in a corresponding way.

15. The method of Claim 14, wherein said order is indicative of whether said data entry is a least recently used entry with respect to other entries.

16. The method of Claim 13, wherein said first bit is set to indicate that an associated way contains non-temporal data.

17. The method of Claim 13, wherein said first bit is cleared to indicate that an associated way contains temporal data.

18. The method of Claim 14, further comprising providing a cache control logic coupled to said at least one cache memory and said processor, for controlling said at least one cache memory.

19. The method of Claim 18, wherein said processor receives an instruction for accessing data, said processor determining if said data is located in said at least one cache memory, if so, accessing said data from said at least one cache memory, otherwise, accessing said data from said main memory.

20. The method of Claim 19, wherein if said data is accessed from said at least one cache memory, said cache control logic determines if said data is temporal, if so, updating an order of said second bit corresponding to said way that is being accessed, otherwise leaving said order unchanged.

21. The method of Claim 19, wherein said first bit corresponding to said way is unchanged.

22. The method of Claim 19, wherein if said data is accessed from said at least one cache memory, said cache control logic configures said first bit to indicate that said accessed data is non-temporal, said cache control logic further updating said order of said second bit.

23. The method of Claim 19, wherein if said data is accessed from said main memory, said cache control logic determines if said data is non-temporal, if so, configuring said first bit to indicate that said

accessed data is non-temporal, said cache control logic leaving unchanged said order of said second bit.

24. The method of Claim 23, wherein if said cache control logic determines that said data is temporal, said cache control logic configures said first bit to indicate that said accessed data is temporal, said cache control logic updating said order of said second bit.

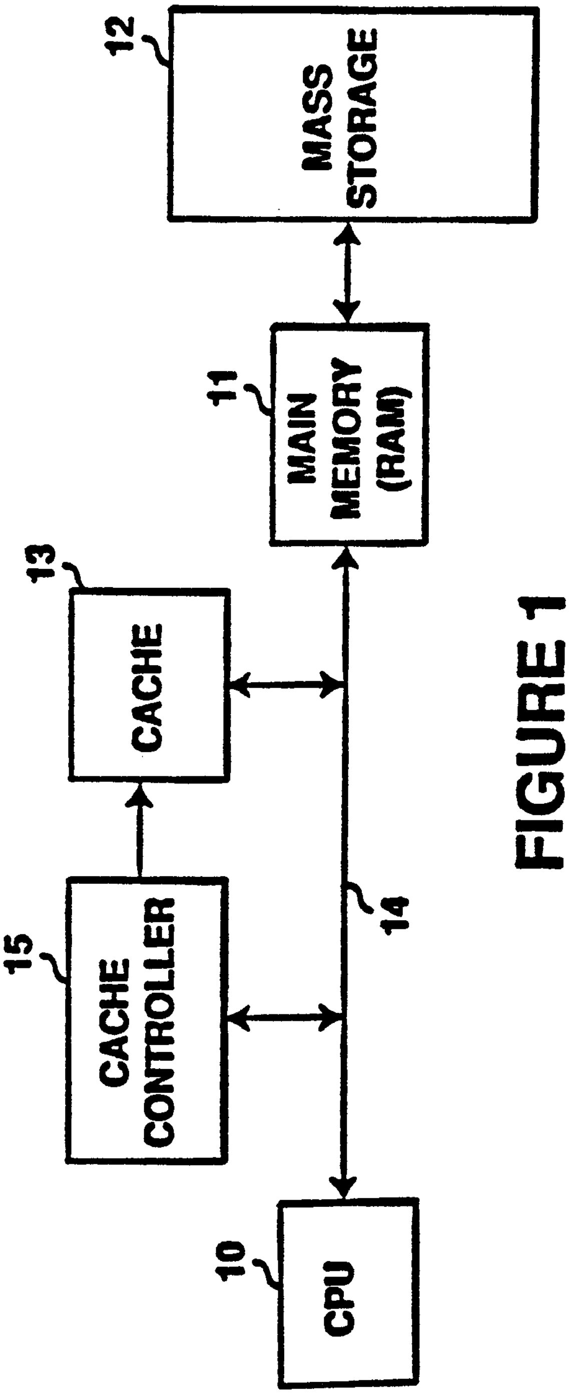

12
MASS STORAGE
11
MAIN MEMORY (RAM)
13
CACHE
15
CACHE CONTROLLER
14
10
CPU

FIGURE 1

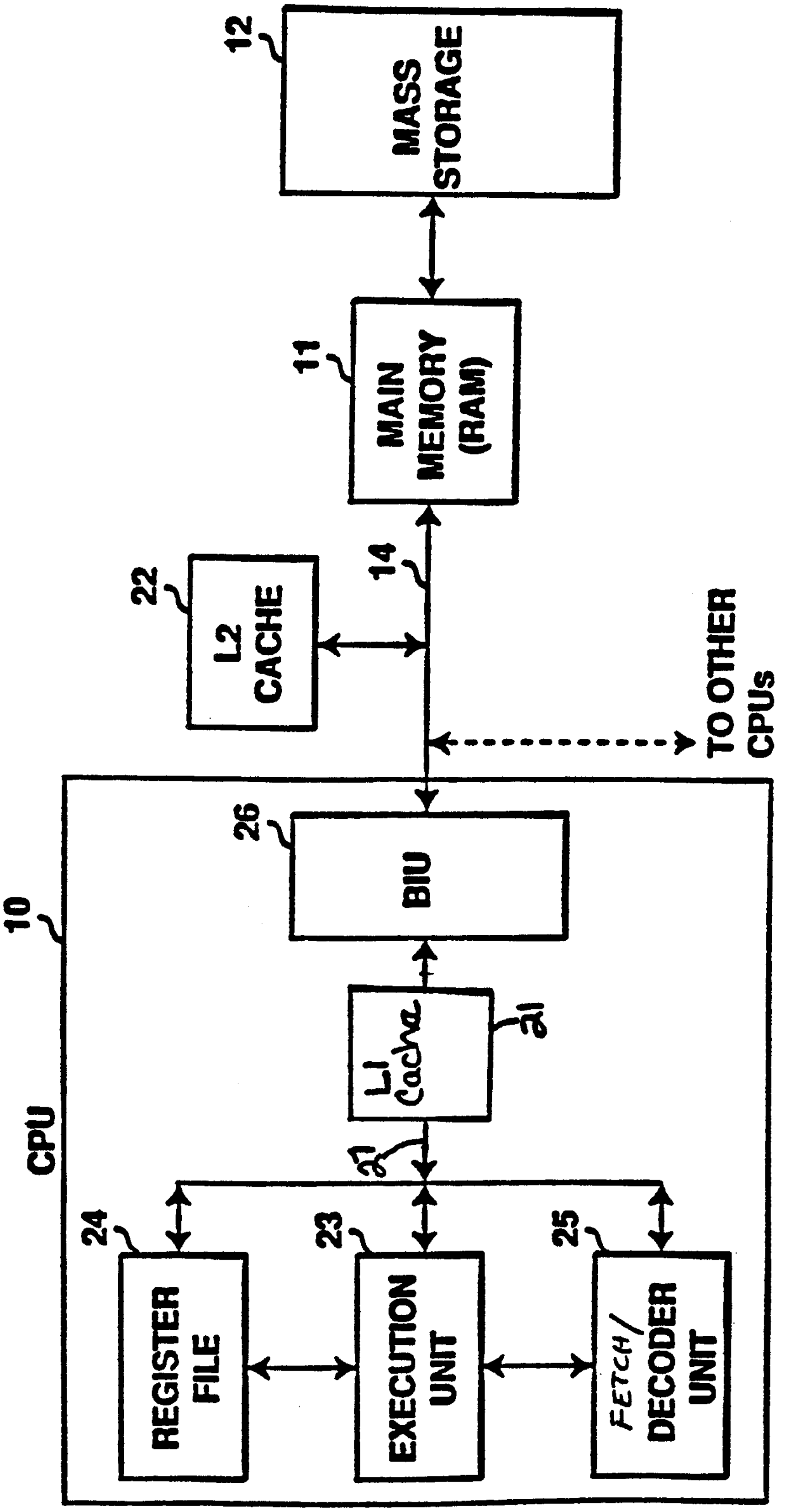

12
MASS STORAGE
11
MAIN MEMORY (RAM)
22
L2 CACHE
14
TO OTHER CPUs
10
CPU
26
BIU
L1 Cache
21
27
24
REGISTER FILE
23
EXECUTION UNIT
25
FETCH/ DECODER UNIT

FIGURE 2

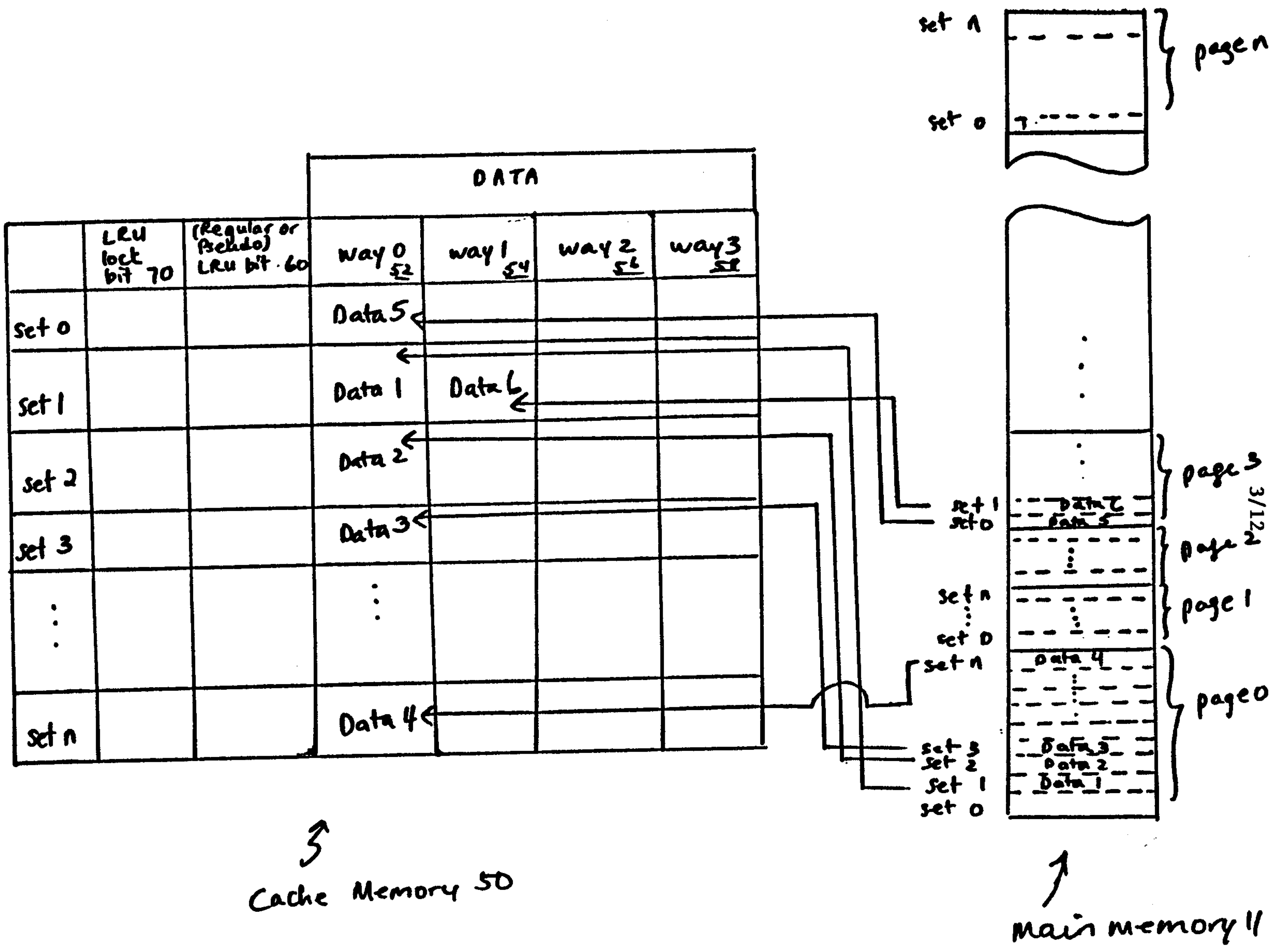

DATA
LRU lock bit 70
(Regular or Pseudo) LRU bit 60
way 0 52
way 1 54
way 2 56
way 3 58
set 0
set 1
set 2
set 3
set n
Data 5
Data 1
Data 6
Data 2
Data 3
Data 4
Cache Memory 50
set n
set 0
page n
page 3
page 2
page 1
page 0
set 1
set 0
set n
set 0
set n
set 3
set 2
set 1
set 0
Data 6
Data 5
Data 4
Data 3
Data 2
Data 1
Main memory 11

Figure 3

	Cache Access Status	Instruction Type	Mode bit status	LRU lock bit (70) status prior to access	LRU lock bit (70) status after cache access	(Regular or Pseudo) LRU bit 60 status
1	Hit	T	X	0	0 (not updated)	Updated according to LRU technique used
2	Hit	T	X	1	1 (not updated)	Not updated
3	Hit	NT	0	0 or 1	Not updated	Not updated
4	Hit	NT	1	0 or 1	1	Updated to the way that was hit
5	Miss	T	X	0 or 1	0	Updated according to LRU technique used
6	Miss	NT	X	0 or 1	1	Not updated (since LRU bit 60 pointsto the way to be replaced, and it is not updated, it will point to the non-temporal data.)

FIGURE 4

X: Don't Care

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	0	way 0				
set 1 . .						

FIGURE 5A

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	0	way 3				
set 1 . .						

FIGURE 5B

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	1	way 0				
set 1 . .						

FIGURE 6A

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	1	way 0				
set 1 . .						

FIGURE 6B

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	0	way 0				
set 1 . .						

FIGURE 7A

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	0	way 0				
set 1 . .						

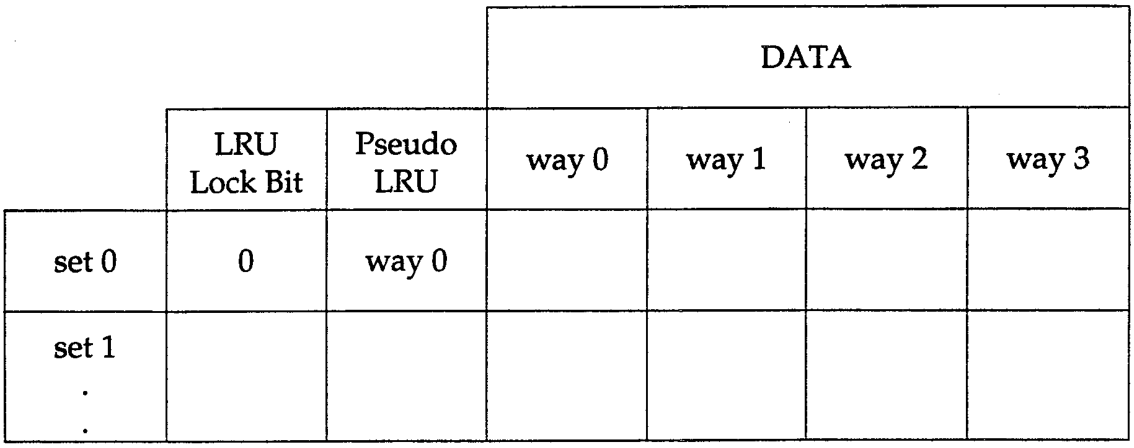
FIGURE 7B

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	1	way 0				
set 1 . .						

FIGURE 7C

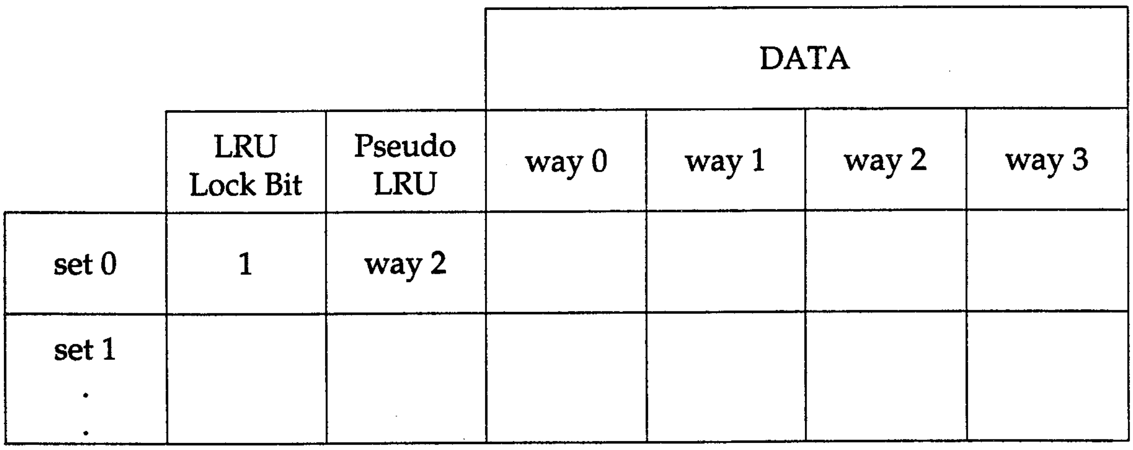
			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	1	way 0				
set 1 . .						

FIGURE 7D



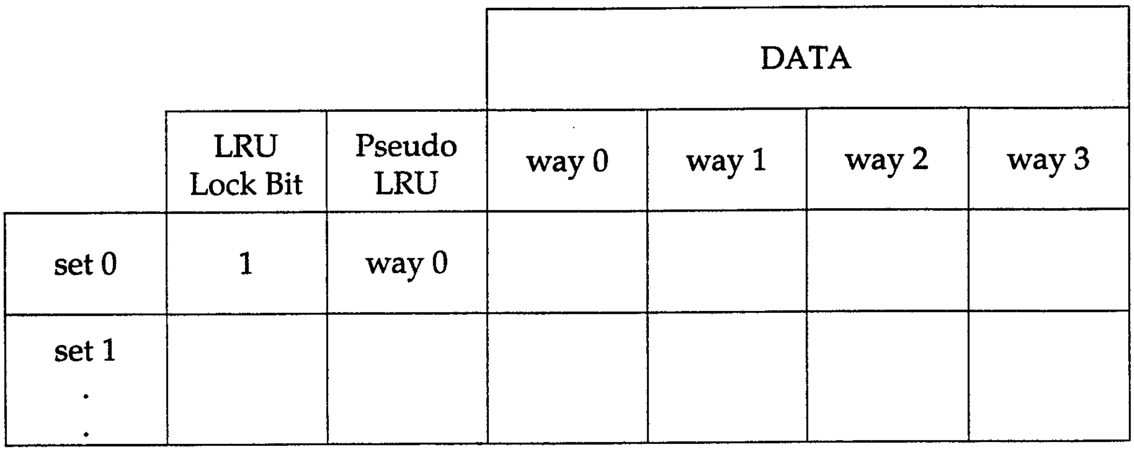
			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	0	way 0				
set 1 . .						

FIGURE 8A



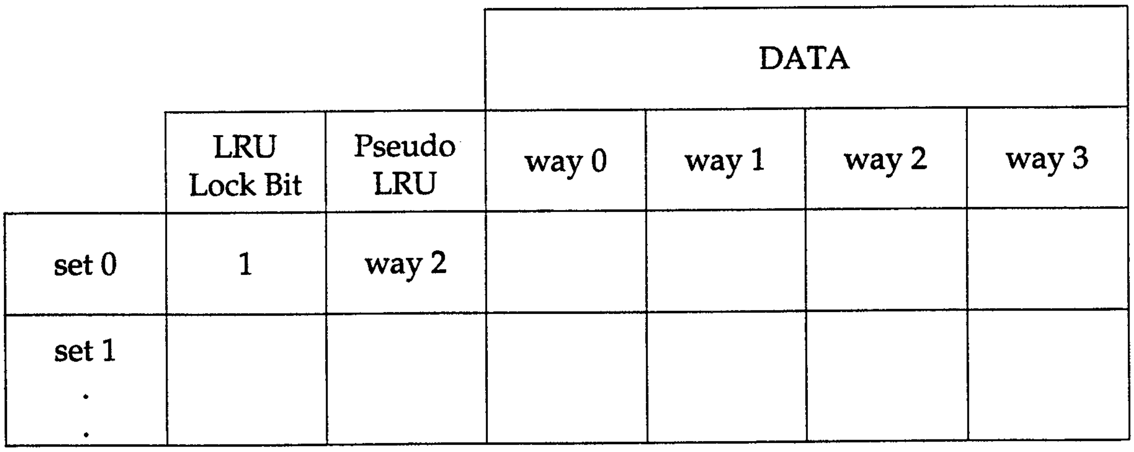
			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	1	way 2				
set 1 . .						

FIGURE 8B



			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	1	way 0				
set 1 . .						

FIGURE 8C



			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	1	way 2				
set 1 . .						

FIGURE 8D

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	1	way 0				
set 1 . .						

FIGURE 9A

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	0	way 3				
set 1 . .						

FIGURE 9B

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	0	way 0				
set 1 . .						

FIGURE 10A

			DATA			
	LRU Lock Bit	Pseudo LRU	way 0	way 1	way 2	way 3
set 0	1	way 0				
set 1 . .						

FIGURE 10B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/06501

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 12/08
US CL :711/128, 129, 133, 136, 154, 160
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 711/128, 129, 133, 136, 154, 160

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS (USPATENT, EPO ABSTRACT, JPO ABSTRACT)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,493,667 A (HUCK ET AL.) 20 February 1996, abstract and Fig. 5.	1-24
Y	US 5,353,425 A (MALAMY ET AL.) 04 October 1994, abstract and Fig. 6.	1-24
Y	US 4,905,141 A (BRENZA) 27 February 1990, col. 3 lines 59-61.	1-24
Y	EP 0 496 439 B1 (CHI, HUNG) 21 January 1998, col.4 lines 43-53.	1-24
Y	US 5,471,605 A (RUBY) 28 November 1995, abstract and Fig. 7.	1-24
Y,P	US 5,829,025 A (MITTAL) 27 October 1998, abstract and Fig. 3.	1-24

☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search 27 MAY 1999	Date of mailing of the international search report 16 JUN 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer HONG C. KIM Joni Hill
Facsimile No. (703) 305-3230	Telephone No. (703) 305-3835

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/06501

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,P	US 5,875,465 A (KILPATRICK ET AL.) 23 February 1999, abstract and Fig. 11.	1-24
Y,P	US 5,845,317 A (PAWLOWSKI) 01 December 1998, abstract and Fig. 1.	1-24
Y, P	US 5,826,052 A (STILES ET AL.) 20 October 1998, abstract.	1-24
Y	US 5,434,992 A (MATTSON) 18 July 1995, abstract.	1-24